Tools for Making Better Architecture Decisions

Ian Gorton
Computational and Information Sciences
Pacific Northwest National Lab

Outline

- Why is architecture difficult?
- ArchDesigner
- BRedB
- Model-driven performance engineering
- Middleware for data intensive computing
Why is architecture hard?

Architecture design typically takes place at an early stage of the project:
- Hard, often impossible, to thoroughly reason about the consequences of many design decisions
- Involves making design decisions that are difficult/costly to change downstream if they are discovered to be flawed
- Complex design trade-offs needed to meet competing architectural requirements
- Put very simply – architecture aims to address any issues that will be expensive/impossible to change once the project progresses

"...The life of a software architect is a long (and sometimes painful) succession of sub-optimal decisions made partly in the dark..."
(Philippe Kruchten)
Specific issues

- **Design**
  - Multiple (conflicting) stakeholder requirements
  - Large number of design options
  - Complex technical issues, design trade-offs, technology selections

- **Analysis and Documentation**
  - Support easier/cheaper downstream modification
  - Capture design rationale behind design decisions
  - Survive changes in staff or outsourced development

- **Risk mitigation**
  - Will system perform/scale to meet requirements?
  - How much hardware needed?
  - Avoid expensive rework required late in project

Architecture Design
Architecture Design Issues

- Typical architecture design scenario
  - numerous design decisions with multiple design alternatives
  - Each decision requires evaluation and selection from among the alternatives
  - Many design alternatives conflict
  - Multiple stakeholders with competing priorities
  - Designs constrained by project cost and schedule
  - Project costs in multi-million $$ range

- We’ve created ArchDesigner to address such scenarios.

ArchDesigner Approach

- Help architects select optimal design alternatives given:
  - Constraints
  - Dependencies
  - Stakeholder priorities

- Exploits management science and operational research techniques:
  - Integer Programming
  - AHP
ArchDesigner Case Study

- Upgrading a major packaged business system (CRM)
- Package can run on:
  - Different operating systems
  - Operating systems runs on different servers/processors
  - Oracle required, only certified on certain OS/CPU/server alternatives
  - 4 vendors, different costs/solutions/support
- 3 key stakeholders

Stakeholder Participation
Step 1: Value Score Computation

- **Value Score** of design alternative X is the degree to which alternative X satisfies desired quality attributes.

**Example:**
- if Performance (P) is "moderately more important" than Modifiability (M) then we have $(P,M)=3$ and $(M,P)=1/3$

![Value Score Diagram](image)

- **Decision Options**: OS, Solaris, XP, Linux
- **QAs**: Cost, Reliability, Manageability, Cluster support

Step 2 and 3

- **Normalize and weight decisions**

- **Use integer programming to select optimal set of design alternatives** subject to
  - Cost
  - Schedule
  - dependencies

![Integer Programming Diagram](image)
Scoring examples

<table>
<thead>
<tr>
<th>Architectural Decision</th>
<th>Alternative</th>
<th>Value Score</th>
<th>Normalized Value Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 D_SERV</td>
<td>1 SMP-Based</td>
<td>0.206</td>
<td>1.379</td>
</tr>
<tr>
<td></td>
<td>2 Commodity-Level</td>
<td>0.704</td>
<td>4.224</td>
</tr>
<tr>
<td></td>
<td>3 Linux</td>
<td>0.283</td>
<td>2.39</td>
</tr>
<tr>
<td></td>
<td>4 SUSE</td>
<td>0.154</td>
<td>1.25</td>
</tr>
<tr>
<td>2 D_OS</td>
<td>1 Solaris</td>
<td>0.522</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>2 HPUX</td>
<td>0.283</td>
<td>2.39</td>
</tr>
<tr>
<td></td>
<td>3 Windows</td>
<td>0.066</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>4 SUSE</td>
<td>0.154</td>
<td>1.25</td>
</tr>
<tr>
<td>3 D_PROC</td>
<td>1 AMD-32</td>
<td>0.532</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>2 AMD-64</td>
<td>0.372</td>
<td>2.69</td>
</tr>
<tr>
<td></td>
<td>3 Intel-32</td>
<td>0.078</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>4 Sunzio-64</td>
<td>0.170</td>
<td>1.19</td>
</tr>
<tr>
<td></td>
<td>5 Ultrisys-64</td>
<td>0.111</td>
<td>0.83</td>
</tr>
<tr>
<td>4 D_DB</td>
<td>1 Oracle 9</td>
<td>0.666</td>
<td>4.40</td>
</tr>
<tr>
<td></td>
<td>2 Oracle 11g</td>
<td>0.637</td>
<td>4.59</td>
</tr>
<tr>
<td></td>
<td>3 Oracle 11g</td>
<td>0.637</td>
<td>4.59</td>
</tr>
<tr>
<td>5 D_VNDIR</td>
<td>1 Vendor-1</td>
<td>0.447</td>
<td>2.99</td>
</tr>
<tr>
<td></td>
<td>2 Vendor-2</td>
<td>0.108</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>3 Vendor-3</td>
<td>0.096</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>4 Vendor-4</td>
<td>0.209</td>
<td>1.49</td>
</tr>
</tbody>
</table>

Decision-centric analysis
Combination-centric analysis

**select best 10 for all stakeholders for all quality attributes**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Server</th>
<th>Oracle DB</th>
<th>Vendor</th>
<th>Processor</th>
<th>OS</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-64</td>
<td>SUSE</td>
<td>14.411</td>
</tr>
<tr>
<td>2</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>Itanium-64</td>
<td>HPUX</td>
<td>14.388</td>
</tr>
<tr>
<td>3</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-32</td>
<td>SUSE</td>
<td>14.334</td>
</tr>
<tr>
<td>4</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-64</td>
<td>RHE</td>
<td>14.331</td>
</tr>
<tr>
<td>5</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-32</td>
<td>RHE</td>
<td>14.254</td>
</tr>
<tr>
<td>6</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-64</td>
<td>Windows</td>
<td>13.947</td>
</tr>
<tr>
<td>7</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-1</td>
<td>AMD-32</td>
<td>Windows</td>
<td>13.870</td>
</tr>
<tr>
<td>8</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-4</td>
<td>AMD-64</td>
<td>SUSE</td>
<td>13.701</td>
</tr>
<tr>
<td>9</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-4</td>
<td>AMD-32</td>
<td>SUSE</td>
<td>13.704</td>
</tr>
<tr>
<td>10</td>
<td>Commodity</td>
<td>Itig</td>
<td>Vendor-4</td>
<td>AMD-64</td>
<td>RHE</td>
<td>13.701</td>
</tr>
</tbody>
</table>

What-If analysis

- ArchDesigner makes it easy to do what-if analysis:
  - assume all vendors rated equal, are design decisions same/similar?
  - ignore cost constraints to find best technical choice
  - Reduce budget to find best/cheaper technical choice
  - Remove low-weight design decisions (sensitivity analysis)

- Eventually 5 viable alternatives emerged:
  1. Vendor1 – Itanium-64 – HPUX.
  2. Vendor4 – AMD-64 – SUSE.
  3. Vendor2 – AMD-64 – SUSE.
  4. Vendor1 – AMD-64 – SUSE.
  5. Vendor2 – Ultrasparc-64 – Solaris.
Benefits

▶ A design decision aid
  - With web-based tool support and query language

▶ Enables architects to:
  - systematically evaluate design alternative combinations
  - consider all possible options by identifying and quantifying all possible permutations
  - conduct what-if and sensitivity analysis
  - explicitly capture design decisions and rationale
  - visibility of decisions to stakeholders/management

▶ If you have a nasty tender to evaluate, challenging stakeholders, complex designs trade-offs ....
Capturing architecture knowledge

- An architecture embodies crucial design decisions
  - Rarely captured in architecture docs (if they exist!)
- If rationale behind design decisions is lost:
  - System evolution becomes hard
  - Difficult to identify design errors
- We did a survey:
  - 80% can’t understand designs without adequate docs
  - 73% forget why they designed something!
- Impediments to capturing design info:
  - 61% have no time/budget/tools
- And built an architecture knowledge management tool
  - BRedB
Architecture knowledge capture

Modeling Architecture Knowledge
BRedB

Project Collaboration

Knowledge-Based
- The knowledge-based involves capturing various knowledge artifacts that users may create, modify, view and search. These artifacts include the following:
  - General Scenario
  - Patterns
  - Analysis Models
  - Architecturally Significant Requirement

Project-Based
- The project-based involves building databases of artifacts for specific projects to support their software architecture evaluation. This can be archived through creating new artifacts and archiving existing artifacts. These new artifacts will be added to the knowledge repository. The project-based artifacts include:
  - General Scenario
  - Patterns
  - Analysis Models
  - Architecturally Significant Requirement
  - Quality Factor
  - Architecture Decisions
  - Alternative Decisions
  - Finding

Search
- Search provides essential functionalities to allow users to search for their desired artifacts. The two different types of search are:
  - Field-based
  - Keyword-based

BRedB Case Study

- Australian Defense Science Technology Organization (DSTO)
  - Avionics architecture assessment
  - Long lived, high cost projects

- BRedB used for avionic architecture evaluation
  - Quality attributes
  - Evaluation framework
  - Risk management
Utility Tree in BRedB

Exploiting General Scenarios

- Domain-specific general scenarios captured
  - increase evaluation effectiveness
  - Easily extensible
  - Represent expert evaluation knowledge
**Reporting**

- Automatically generate evaluation reports
  - Shows how architectural decisions satisfy a concrete scenario

**Initial Findings**

- BRedB added considerable rigour to the DSTO evaluation process
  - Repository of expert knowledge in general scenarios
  - Archive of past project experience valuable over time
  - Effective mechanism to organise and query large amounts of architecture knowledge

- BRedB should be useful if you:
  - are outsourcing/off-shoring/purchasing your systems?
  - have superhero architects prone to sudden departure or large pay increase demands

- More studies and R&D needed
  - We're keen to work with interested partners …
Architecture Risk Mitigation

SOA Performance Engineering

- Service-Orientations Architecture (SOA) being widely adopted
  - SOAs built using J2EE/.NET/Web Services technologies
  - Need to be scalable and reliable
- Performance/scalability issues are complex
  - Application code accounts for small percentage of execution time
  - Middleware, databases, Web servers, etc
- Many applications fail disastrously due to lack of attention to performance
Web Services Performance

- A common concern due to XML
  - Verbosity
  - Processing overheads
- What’s the reality?
- Some experiments performed at CSIRO’s ICT Centre in 2005

Performance Prototyping

- Performance prototyping is time-consuming and tricky
  - Build server prototype (if no existing system)
  - Configure server/database, test prototype
  - Load test data
  - Build test client
    - Multithreaded
    - Generate variable loads and test data
  - Instrument client and server to capture performance
  - Capture/process/analyze results
  - Fix the stuff you got wrong and run tests again … and again …
- We’re trying to make this easy …
Performance Modeling Process

Model-Driven Performance Engineering

Use case diagram
Sequence diagram
Deployment diagram

Application Model
Architecture Model
Networking Topology
Queuing Network Model

MDApert: An eclipse plug-in for performance prediction
MDAbench/DSLBench: UML/Microsoft DSL benchmark generation engine
MDABench

MDABench – example outputs
DSL v MDA

- MDA is an OMG driven initiative for model-driven development
  - Based on UML models and tools
- DSL is Microsoft’s model-driven development initiative
  - If you don’t like/need UML, roll your own modeling language

DSL Bench: Meta-level Modeling

![Diagram showing meta-level modeling with components like Client, Coordinator, Debug, ResultCollector, DataGenerator, and others.](image_url)
Current Status

- Prototype tools deployed in a major case studies
  - Australian Tax Office and Medicare SOA integration
  - SOA for home mortgage process integration
- If you have gnarly, risky SOAs whose performance you don’t quite understand ....

Middleware for Data Intensive Computing
Data Intensive Computing

Data-intensive computing is managing, analyzing, and understanding data at volumes and rates that push the frontier of current technologies.

- Gb/s – Tb/s data streams
- Heterogeneous data
- Distributed resources for data, computing
- Terabyte to petabyte data volumes
- Complex non-linear analysis

Analytical Processing Pipelines

- Common design pattern requires:
  - Flexible integration
  - Ability to process massive data volumes
  - High performance and scalability
Analytical Processing Pipelines

MeDICI

Adaptive Workflow

Data Virtualization

SIFT

Grid Computing

Workstations

High-Performance Computers

Data Resources

Data Content Locations

Preprocessing Metadata Store
MeDICI Status

- Using MeDICI to design/build several large scale data intensive applications:
  - Cyber-security
  - Bio-informatics
  - Proteomics
- Initial API and programming guide coming this summer ...
- Open source release is in the plans for later in 2007

That’s it ...

- Paying attention to your architecture will provide large returns on investments
- Upfront effort is needed
  - “no free lunch”, but methods and tools will help
  - remember, you pay now or later, just a lot less now!!
- Advanced technologies can make a difference
  - We’re keen to work with organizations to trial and deploy
Questions?

Some answers …